Intelligence and Behavioral Boundaries

Scott A. Wallace and John E. Laird
Artificial Intelligence Laboratory
University of Michigan
Ann Arbor, MI 48109, USA
{swallace,laird}@umich.edu

ABSTRACT

In this paper, we examine Newell's definition of an agent's intelligence. The definition implies a very strict correlation between the agent's knowledge, its mind, and its behavior. Based on this correlation we argue that a concise and well–formed representation for the agent's behavior is often essential for measuring its intelligence. To meet this need, we present Behavioral Bounding, a method that can be used to calculate a concise, high–level representation of behavior. We examine both the theoretical commitments of this approach, as well as practical limitations.

KEYWORDS: *version–spaces, behavior representations*

1 INTRODUCTION

In his 1990 book, *Unified Theories of Cognition* (UTC) [6], Newell observes that notions of intelligence are often linked to an underlying service to a particular cause. In this sense, there is no single universal concept of intelligence, but rather a multitude of intelligence measures each related to a particular field, task, or area of interaction. This idea gives rise to such common expressions as *academic intelligence* and *real–world intelligence*. As Newell indicates, the value of an intelligence metric is presumably to help identify which minds can perform which tasks, and what their relative abilities might be.

As a foundation for talking about intelligent agents, Newell introduces the notion of a *knowledge–level* system [5, 6]. As he describes it, such a system is located within an environment and performs a series of actions to obtain its goals; it selects between potential actions by using all of its relevant knowledge [6, pp. 50]. From this concept, Newell constructs the following definition of intelligence: "A system is *intelligent* to the extent that it approximates a knowledge–level system" [6, pp. 90]. That is, a system is perfectly intelligent if it uses all of its available knowledge to achieve its goals. Furthermore, lack of knowledge is distinct from a lack of intelligence. Thus, it often makes most sense to compare the intelligence of two systems with respect to a explicit body of knowledge and a specific problem domain as when one assess the impact of street–smarts on academic activities.

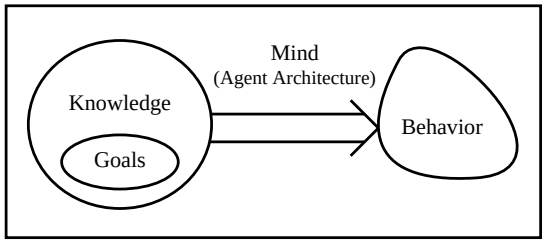

Figure 1: A Knowledge–Level System

Because a true knowledge–level system uses its knowledge to generate behavior that achieves its goals, the system's behavior should be predictable (up to aspects of behavior about which the agent is indifferent) so long as its knowledge is completely known (see Figure 1). A system that approximates the knowledge level, however, is one that cannot reliably make use of all its knowledge. Prediction of this system's behavior requires knowing what the system knows, as well as how its mind works (i.e. the implementation details governing how knowledge is brought to bear, in essence the agent architecture).

The relationship between knowledge, mind and behavior means that given information about two of these three components, it is possible to make a reasonable hypothesis about the third component. In this manner, one can determine how well an agent approximates the knowledge–level, how *intelligent* it is, so long as the agent's knowledge and its behavior are known.

2 BEHAVIOR & INTELLIGENCE

In some situations, it may be difficult or impossible to completely and correctly specify the agent's knowledge. This may be because the agent cannot be trusted to accurately communicate what it knows. Alternatively, it may be because it is exceedingly difficult to interpret the agent's native encoding of knowledge. In either of these two situations, it seems reasonable to rely on the knowledge engineer, who designed the agent, to provide a specification for what the agent knows. Unfortunately, the knowledge engineer's response may be misleading. This is because the agent's knowledge may not have been fully validated and thus may differ from the developers specification (i.e. it may contain errors).

If it is impossible to completely and correctly specify the agent's knowledge, it is also impossible to apply Newell's definition of intelligence in its strict sense. However, we may still profit from an approximate measure of intelligence based on beliefs about what the agent knows. Since these beliefs need not be completely accurate, they should be relatively easy to obtain. Beliefs about the agent's knowledge may come from information offered by the agent itself or from the designer of that agent. Given this approximate information about the agent's knowledge, we can obtain an approximate measure of its intelligence by observing its behavior.

Regardless of whether we have perfect information about the agent's knowledge, making an inference about the agent's intelligence requires the ability to classify and represent behavior in a concise, and well–formed structure. In this paper, we will outline a technique for this purpose. This technique, which we call Behavioral Bounding (B–Bounding), allows us to construct a concise representation of how an agent may perform in various situations. Once this representation has been computed, it can be used to: obtain a true measure of the agent's intelligence (given complete information about the agent's knowledge); to compare relative approximate intelligence (between two agents that are believed to have the same knowledge); or validate an agent's knowledge base (given complete information about the capabilities of the agent's mind). In this paper, we will concern ourselves with the first two applications.

3 MEASURING INTELLIGENCE

As we outlined in Section 2, we can generate an approximate measure of a system's intelligence given a set of beliefs about that system's knowledge and a set of observations about its behavior. The basic process for determining the system's intelligence is given by Algorithm 1.

Algorithm 1 Measure Intelligence

```
B ← Beliefs About Agent's Knowledge
O ← Sequences of Observed Agent Behavior
I ← Max Intelligence
for all observed (state,action) sequence o in O do
  if o can be explained using B then
    do nothing
  else
    decrease I in a meaningful way[1]
  end if
end for
return I
```

This algorithm will return a measure of the agent's intelligence given any observed behavior and any set of beliefs about the agent's knowledge. The exact nature of this algorithm's result, however, is closely tied to the quality of input.

The first parameter of the algorithm is B, a set of beliefs about the agent's knowledge. The accuracy of the measurement will depend on how closely these beliefs reflect the agent's actual knowledge. If, for example, the beliefs turn out to have nothing in common with the agent's actual knowledge, none of the agent's actions will be explainable, and so the agent will be presumed to have a very low (or zero) intelligence, regardless of whether it is a true knowledge–level system. On the other hand, if the beliefs correspond exactly to the agent's true knowledge, then any incorrect behavior will be appropriately ascribed to a lack of intelligence.

The second parameter of the algorithm, O, is a set of $(state_0, action_0) \ldots (state_n, action_n)$ sequences describing behavior the agent was observed performing. This parameter affects the generality of the measurement for the agent's intelligence. If the scope of action sequences spans a large number of tasks and the agent's behavior is diverse, the measurement of intelligence is likely to be more general than if the agent was observed only within a very narrow context. When determining what agent to select for a particular task, one would like the most intelligent agent with respect to the knowledge needed to perform that task. Furthermore, if the task is well specified, the agent's intelligence in areas known to be outside of the

[1] There are many possible methods of decreasing the agent's intelligence score. Naively, one could set $I_{max} = 100$, and for each unexplained action decrease I by $1/N$ where N = *total number of explanations attempted* thus producing a normalized measure. A discussion of the consequences of different approaches, however, is outside the scope of this paper.

task's score can be ignored. On the other hand, if the exact nature of the task is ill–defined, it may be important that the agent have a high general intelligence, even if it is less intelligent on some specific subset of tasks.

The simple algorithm we have presented allows the flexibility to compute measurements of intelligence that span different degrees of accuracy and generality. However, it does suffer from two significant drawbacks if it is used to compute a relatively general measure of intelligence.

Most importantly, Algorithm 1 requires explaining all the actions in each sequence of observed agent behavior. As the generality of the intelligence measure increases, the number of agent traces used by the algorithm will also increase. Since the cost of explaining each observation of behavior is likely to be very high, this could make calculating certain measurements of intelligence infeasible.

In addition, Algorithm 1 does not have any built in method to determine the breadth of behavior that was examined to compute the agent's intelligence. As a result, it may be unclear what types of tasks the measurement pertains to. Furthermore, in cases where the observations of agent behavior are similar, we would like to be able to exploit information acquired during previous explanations.

Improvements to the simple algorithm would reduce the cost of general measures of intelligence, and provide an indication of the space of tasks or situations to which the measurement can be applied. In the remaining sections of the paper, we describe the Behavioral Bounding method, and how it can be used to overcome these shortcomings.

4 BEHAVIORAL BOUNDING

The Behavioral Bounding method can be used to overcome the two faults with the basic *Measure Intelligence* algorithm described in Section 2. The main idea behind the methodology is to construct a concise high–level representation of an agent's behavior from a set of observations. This new representation allows two new methods of measuring general intelligence without explaining all of the behavior in each observation. In addition, the representation itself can be used to obtain a measure of the diversity of behavior encapsulated in the observed agent actions.

Conceptually, Behavioral Bounding is very similar to Mitchell's Version Space framework [4]. Given a language to describe an agent's behavior, and a general to specific ordering over representations in this language, we can construct a maximally specific representation of the observed agent behavior. Figure 2 illustrates this concept. Each

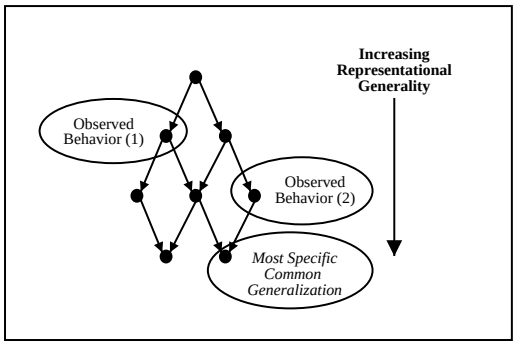

Figure 2: Ordered Behavior Representations

node corresponds to an abstract representation of behavior. Nodes toward the top of the lattice are more specific representations than nodes toward the bottom. Each of the observed agent behaviors is mapped onto this lattice, and the most specific common generalization can then be used to represent all of the behaviors that have been observed. Figure 2 illustrates a specific case with two observations of the agent's behavior. The abstract representation covering both of these observations is their common descendant.

Behavioral Bounding allows two new methods of measuring intelligence, both of which have an improvement in performance over our original simple algorithm. In the first method, the abstract representation of the agent's behavior is computed using all the available observations. Next, we substitute this abstract representation of behavior for the set of observations, O, in the original algorithm and proceed with the calculation in an otherwise normal manner. The benefit of this approach is that the number of explanations is no longer strictly a function of the number of observations. However, in practice this method may be difficult to apply. This is because the induced behavioral representation may be overly general, and as a result may encapsulate behaviors that the agent cannot explain. In such situations, the measurement of the agent's intelligence may be incorrect.

The second variation on our original algorithm overcomes this flaw. This approach uses the generalized behavior representation to tune the simple algorithm without actually substituting it for the observed agent behavior O. As in our first modification to the simple algorithm, we begin by constructing a general representation of the agent's behavior. Just as before, we initialize this representation, R_s, with the most specific representation of all possible behavior (this corresponds to initializing the S–SET in standard version space). Then, each observed behavior trace $o \in O$ is used to iteratively generate a new general representation, R'_s, of o and R_s. In our first variation, this was done

by simply setting $R_s \leftarrow R'_s$ and repeating for all $o \in O$, thereby obtaining a generalization of all the traces in O. In this second approach, however, we examine the differences between R'_s and R_s before proceeding with the generalization. The aspects of R'_s that are more general than R_s indicate aspects of o that must be explained, whereas aspects of behavior that remain unchanged between R_s and R'_s have already been explained in a previous iteration. In this way the modified algorithm performs the minimal amount of explication for any set of observations.

The second variation of our original algorithm hints as to how the generalized behavior representation can be used to measure the diversity of observed agent behavior. After viewing a single instance of agent behavior, R_s will be generalized to the most specific representation that covers that behavior. Given progressively more instances of behavior, R_s will be generalized the minimal amount necessary to cover these observations. A long series of observations in which the agent's behavior is relatively similar will result in very few generalizations. Thus, differences (in terms of generality) between the most specific behavior representation and the behavior representation that covers all of the observations provides an automatic way of determining the generality of the intelligence measurement.

5 IMPLEMENTING B–BOUNDING

At a theoretical level, Behavioral Bounding, has very few requirements. All that is needed is a language capable of representing the agent's behavior and an ordering from specific to general over potential behavioral descriptions in this language. However, in order to make B–Bounding practical, a number of other constraints must be met. Most importantly, the language used to represent an agent's behavior must simultaneously be rich enough to distinguish between appropriate and inappropriate behavior, while also being constrained enough so that the learning problem remains tractable.

5.1 POTENTIAL REPRESENTATIONS

In order to determine a suitable language for describing agent behavior at a high–level, we should begin by examining the representation of a single observation of behavior. A single observation of behavior can be described by the list of ordered pairs $((S_1, B_1), (S_2, B_2), \cdots, (S_n, B_n))$ where ordered pair (S_i, B_i) indicates the behavior pursued in the given state. (Note that we can guarantee the uniqueness of the S_i by including the value of a world clock in the state description). At a minimum, each of the B_i encodes the external actions performed by the agent,

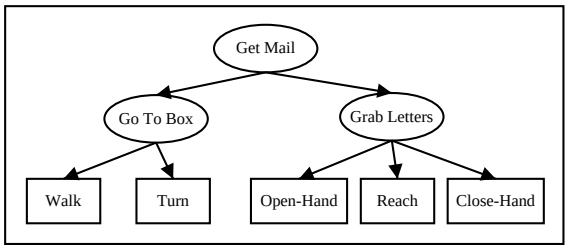

Figure 3: Hierarchical Goal Structure

but in some cases, B_i may contain additional information about the agent's behavior.

In the minimal case, where B_i contains only the externally observable action pursued by the agent, it is possible to abstractly represent the space of behaviors with canonical forms. Using this approach, states and actions are grouped into equivalence classes. For example, two distinct actions *Turn Right Quickly* and *Turn Right Slowly* might be grouped into a more generic *Turn Right* class. Similarly, particular features of the state space may be ignored within a portion of the problem domain yielding a set of equivalent states. For example, minor deviations from normal summer–time temperatures are unlikely to impact the manner in which an airplane is controlled.

When abstraction is used to reduce the complexity of the overall problem, each state and action in the observed behavior is replaced with the canonical form that represents the equivalence class of the observable. This representation has the ability to greatly reduce the size of the behavioral representation space, but this ability comes at a high cost. Using this method requires constructing these equivalence classes based on properties of the task that the agent is performing. Unfortunately, it is unlikely that an automated approach would be able to correctly identify these equivalence relations through direct examination of the environment. Instead, it is much more likely that this job would fall upon a human designer. Any additional human effort will increase the cost of performing the intelligence test, but constructing equivalence classes may be particularly costly because it must be done each time the test is performed in a new environment.

A less costly method of constructing a high–level representation of behavior can be performed if we assume the ability to gather more information about the agent's behavior. In particular, if we assume that the behavioral descriptions B_i contains both the agent's external actions as well as the agent's motivation for performing that action (i.e. the agent's current goals), we can use a high–level description of the agent's behavior without constructing a set of equivalence classes over states and actions.

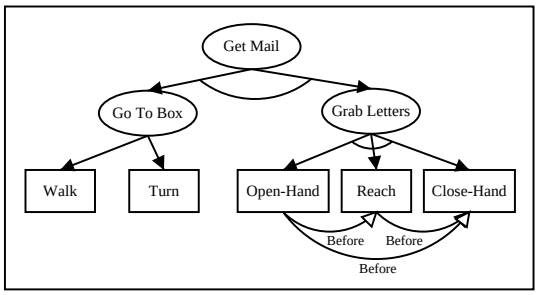

Figure 4: Constrained Goal Structure

5.2 EXPLOITING GOAL STRUCTURE

Because an agent's knowledge is structured around goals and actions, development of the agent's knowledge base is likely to profit from an explicit representation of these relationships. We will call this representation a goal hierarchy (see Figure 3) and it can be used to define a meaningful structure for the agent's knowledge similar to the way an outline serves to help structure the ideas in an article. The inner nodes in the goal hierarchy correspond to goals, while the leaves correspond to primitive actions. A node's descendants are the sub–goals and actions that may help to achieve the goal represented by the specified node.

The relationships described in the goal hierarchy make it possible to identify the set of primitive actions one would expect to observe from an agent pursing any particular goal. Thus, according to behavior represented by Figure 3, one would expect that an agent pursuing the goal of *Go To Box* would use the primitive actions *Walk* and *Turn*. A goal hierarchy in this form can be viewed as a general specification for behavior. By defining a set of constraints that can be added to the hierarchy, we can increase the specificity of the behavioral representation. This new language, formed by the goal hierarchy and constraints that can act upon it, can then be used by the B–Bounding technique to generate a concise representation of an agent's behavior.

To determine an appropriate set of constraints, we look to previous work in Hierarchical Task Networks (HTNs) [3, 1] from the planning community and Goal, Operator, Method and Selection–Rule (GOMS) models [2] from the HCI community.

HTNs consist of two types of nodes: goals and primitive actions. Methods indicate how goals decompose into sequences of sub–goals (or at the lowest level of abstraction, into primitive actions). Clearly, the HTN model has much in common with the goal hierarchy described above. However it also expands upon the basic structure in two ways:

1. HTNs explicitly represent alternative ways of decomposing a goal into sub–goals.
2. HTNs are able to explicitly represent ordering constraints between siblings nodes (so, for example, a method can describe a specific order in which the sub–goals must be accomplished).

GOMS models also consist of two types of nodes: goals and operators, which map directly onto the primitive actions of HTNs. Methods in a GOMS model performs the same function as in HTNs, and it is assumed that these methods have the ability to represent ordering constraints on the nodes in the decomposition. The main difference between HTNs and GOMS models is that the later use explicit structures (selection–rules) to determine the set of methods that are appropriate for a particular situation.

Our representation language is constructed from the common components that underlie both HTNs and GOMS models. Specifically, we have two distinct types of constraints: goal types and ordering constraints. Goals can be either of two types AND or OR. An AND node indicates that in order to accomplish the goal represented by the node, all of the children must be accomplished. An OR node indicates that one (or more, but not all) of the children must be accomplished in order to accomplish the goal. By using multiple levels of the hierarchy AND and OR nodes can be used to represent alternative *methods* for solving a higher–level goal, just as in HTNs and GOMS models. The second type of constraint in our model indicates the valid orders in which goals or actions can be accomplished. For this purpose, we use binary temporal constraints, to build a partial ordering between the children of each goal node. Figure 4 illustrates a goal hierarchy that is partially constrained. In this example, the constraints indicate:

- Achieving the goal *Get Mail* requires accomplishing both of the sub–goals *Go To Box* and *Grab Letters*.
- Achieving the goal *Grab Letters* requires performing the actions *Open Hand*, *Reach*, and *Close Hand* in that order.

With these two types of constraints we can specify a general to specific ordering over abstract behavior representations. In this ordering, the maximally general behavior specification is one in which all nodes are of type OR, and no temporal constraints exist. The maximally specific behavior representation in which all goals are of type AND and there is a total ordering between all siblings. Because these constraints underlie both GOMS models and HTNs, we can be confident that they will provide a good medium

for representing behavior in a variety of domains. Furthermore, because the goal–hierarchy itself can be used as an organizational tool to help outline and develop the agent's knowledge, it is likely that this representation can be used with little cost.

6 CONCLUSION

The definitions put forth in Newell's book *Unified Theories of Cognition* provide a basis for constructing a simple algorithm to measure the intelligence of a particular system. However, a straightforward implementation of this metric may require impractical computational resources when a measurement of general intelligence is required. We have presented the Behavioral Bounding method that mitigates this problem by ensuring that the minimal amount of computation is performed to gain a measurement of an agent's intelligence. In addition, Behavioral Bounding can be used to explicitly indicate the relative generality of a particular intelligence measurement, thus overcoming another flaw in the original algorithm without incurring additional cost.

Recently, our research has been focusing on using the Behavior Bounding method to validate an agent's knowledge based on observations of its behavior (the third application of B–Bounding outlined in Section 2). We have implemented and begun an investigation of Behavioral Bounding using the goal hierarchy and constraint language discussed in Section 5.2. Future work will focus on enriching the constraint language and new uses for this technique.

REFERENCES

[1] Kutluhan Erol, James Hendler, and Dana S. Nau. UMCP: A sound and complete procedure for hierarchical task–network planning. In *Artificial Intelligence Planning Systems*, pages 249–254, 1994.

[2] Bonnie E. John and David E. Kieras. The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer–Human Interaction*, 3(4):320–351, 1996.

[3] Subbarao Kambhampati. A comparative analysis of partial order planning and task reduction planning. *SIGART Bulletin*, 6(1):16–25, 1995.

[4] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.

[5] Allen Newell. The knowledge level. *AI Magazine*, 2(2), 1981.

[6] Allen Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, Mass, 1990.